4.5 X-ception, ResNet50, Inceptionv3

Data Preprocess of X-ception, ResNet50, Inceptionv3

In [ ]:
```
import os
import numpy as np
from os import listdir
from imageio import imread
from keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from keras.utils.image_utils import img_to_array

import PIL
import matplotlib.pyplot as plt
```

In [ ]:
```
# Settings
num_classes = 10
test_size = 0.2
```

read image and convert to 3d array

In [ ]:
```
def get_img(data_path):
  ## Getting image array from path:
  img = PIL.Image.open(data_path)
  img = img.convert("L")
  img = img_to_array(img)
  img = np.resize(img, (100, 100, 3))
  return img
```

Get dataset from picture and then split to train and test set

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
dataset_path = "/content/drive/MyDrive/Dataset"

## Getting all data from data path
labels = sorted(listdir(dataset_path))
X = []
Y = []
for i, label in enumerate(labels):
  data_path = dataset_path + "/" + label

  for data in listdir(data_path):
    img = get_img(data_path + "/" + data)
    X.append(img)
    Y.append(i)
## create dataset
X = 1 - np.array(X).astype("float32") /255
Y = np.array(Y).astype("float32")
Y = to_categorical(Y, num_classes)

X, X_test, Y, Y_test = train_test_split(X, Y, test_size=test_size, random_state = 42)
print(X.shape)
print(X_test.shape)
print(Y.shape)
print(Y_test.shape)
```

```
Mounted at /content/drive
(1649, 100, 100, 3)
(413, 100, 100, 3)
(1649, 10)
(413, 10)
```

In [ ]:
```
import tensorflow as tf
from numpy.random import seed
seed(123)
tf.random.set_seed(123)
```

In [ ]:
```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import pandas as pd
import sklearn as sk
import time
from keras.datasets import mnist
from keras.models import Sequential, load_model
from keras.layers import Dense, Dropout, Flatten
from keras import optimizers
from keras import backend as K
from keras import regularizers
from keras import initializers
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
import math
from keras import applications
```

In [ ]:
```
img_height = 100
img_width = 100
```

In [ ]:
```
# Creating validation set and training set by partitioning the current training set
val = X[:274]
partial = X[274:]
val_labels = Y[:274]
partial_labels = Y[274:]
```

In [ ]:
```
print(X.shape)
print(val.shape)
print(partial.shape)
```

```
(1649, 100, 100, 3)
(274, 100, 100, 3)
(1375, 100, 100, 3)
```

X-ception

When building the last layers of X-ception, I first added the GlobalAveragePooling2D() to create feature map for each cagetory. I then added dense layer but it didn't help. I tried several drop out values and found 0.4 the best. After tuning the last layers, I unfreeze the base model and retrain the whole model with a very low learning rate. I've tried some different values of learning rate and found lr = le-5 the best. When fit the model, I used EarlyStopping function in keras to find the optimal epoch value (=27) to avoid the issue of overfiffting.

In [ ]:
```
#Load the Xception pre-trained model
#include_top=False means that you’re not interested in the last layer of the model. You
base_model = keras.applications.Xception(
    weights='imagenet',
    input_shape=(img_height, img_width, 3),
    include_top=False)
```

In [ ]:
```
#To prevent the base model being retrained
base_model.trainable = False
```

In [ ]:
```
inputs = keras.Input(shape=(img_height, img_width, 3))
```

In [ ]:
```
#Preprocess inputs as expected by Xception
#scale from (0,1) to (-1,1)
x = tf.keras.applications.xception.preprocess_input(inputs)
```

In [ ]:
```
#Build the last layers
#Use the functional API method in Keras to illustrate this approach
x = base_model(x, training=False)
x = keras.layers.GlobalAveragePooling2D()(x)
x = keras.layers.Dropout(0.4)(x)
outputs = keras.layers.Dense(10)(x)
model = keras.Model(inputs, outputs)
```

In [ ]:
```
model.summary()
```

```
Model: "model_16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_36 (InputLayer)       [(None, 100, 100, 3)]     0         
                                                                 
 tf.math.truediv_17 (TFOpLam  (None, 100, 100, 3)      0         
 bda)                                                            
                                                                 
 tf.math.subtract_17 (TFOpLa  (None, 100, 100, 3)      0         
 mbda)                                                           
                                                                 
 xception (Functional)       (None, 3, 3, 2048)        20861480  
                                                                 
 global_average_pooling2d_16  (None, 2048)             0         
  (GlobalAveragePooling2D)                                       
                                                                 
 dropout_16 (Dropout)        (None, 2048)              0         
                                                                 
 dense_34 (Dense)            (None, 10)                20490     
                                                                 
=================================================================
Total params: 20,881,970
Trainable params: 20,490
Non-trainable params: 20,861,480
_________________________________________________________________
```

In [ ]:
```
model.compile(optimizer='adam',
              loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
model.fit(X, Y, epochs=3, validation_data=(X_test,Y_test))
```

```
Epoch 1/3
52/52 [==============================] - 5s 58ms/step - loss: 2.2971 - accuracy: 0.1055
```

```
- val_loss: 2.2941 - val_accuracy: 0.1259
Epoch 2/3
52/52 [==============================] - 2s 37ms/step - loss: 2.2825 - accuracy: 0.1686
- val_loss: 2.2829 - val_accuracy: 0.1453
Epoch 3/3
52/52 [==============================] - 2s 37ms/step - loss: 2.2684 - accuracy: 0.1740
- val_loss: 2.2713 - val_accuracy: 0.1743
```

Out[ ]:
```
<keras.callbacks.History at 0x7fd436c87d90>
```

In [ ]:
```
# Fine-tuning
base_model.trainable = True
model.summary()

model.compile(
    optimizer=keras.optimizers.Adam(1e-5),  # Low learning rate
    loss=keras.losses.CategoricalCrossentropy(from_logits=True),
    metrics=['accuracy']
)
```

```
Model: "model_16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_36 (InputLayer)       [(None, 100, 100, 3)]     0

 tf.math.truediv_17 (TFOpLam  (None, 100, 100, 3)      0
 bda)

 tf.math.subtract_17 (TFOpLa  (None, 100, 100, 3)      0
 mbda)

 xception (Functional)       (None, 3, 3, 2048)        20861480

 global_average_pooling2d_16  (None, 2048)             0
  (GlobalAveragePooling2D)

 dropout_16 (Dropout)        (None, 2048)              0

 dense_34 (Dense)            (None, 10)                20490

=================================================================
Total params: 20,881,970
Trainable params: 20,827,442
Non-trainable params: 54,528
_________________________________________________________________
```

In [ ]:
```
from keras import callbacks
earlystopping = callbacks.EarlyStopping(monitor ="val_loss",
                                        mode ="min", patience = 5,
                                        restore_best_weights = True)

history = model.fit(partial, partial_labels, batch_size = 16,
                    epochs = 100, validation_data =(val, val_labels),
                    callbacks =[earlystopping])
```

```
Epoch 1/100
86/86 [==============================] - 11s 84ms/step - loss: 2.1110 - accuracy: 0.1927
- val_loss: 1.8732 - val_accuracy: 0.2920
Epoch 2/100
86/86 [==============================] - 7s 85ms/step - loss: 1.7513 - accuracy: 0.3236
- val_loss: 1.7132 - val_accuracy: 0.2993
Epoch 3/100
86/86 [==============================] - 7s 78ms/step - loss: 1.4120 - accuracy: 0.4465
```

```
- val_loss: 1.4318 - val_accuracy: 0.4234
Epoch 4/100
86/86 [==============================] - 7s 77ms/step - loss: 1.2753 - accuracy: 0.4996
- val_loss: 1.2722 - val_accuracy: 0.5182
Epoch 5/100
86/86 [==============================] - 6s 73ms/step - loss: 1.1707 - accuracy: 0.5462
- val_loss: 1.1859 - val_accuracy: 0.5803
Epoch 6/100
86/86 [==============================] - 6s 74ms/step - loss: 1.1123 - accuracy: 0.5862
- val_loss: 1.0707 - val_accuracy: 0.5876
Epoch 7/100
86/86 [==============================] - 6s 73ms/step - loss: 1.0861 - accuracy: 0.5876
- val_loss: 1.2931 - val_accuracy: 0.4708
Epoch 8/100
86/86 [==============================] - 6s 74ms/step - loss: 1.0506 - accuracy: 0.6065
- val_loss: 0.9768 - val_accuracy: 0.6496
Epoch 9/100
86/86 [==============================] - 6s 74ms/step - loss: 0.9626 - accuracy: 0.6429
- val_loss: 1.3748 - val_accuracy: 0.4964
Epoch 10/100
86/86 [==============================] - 6s 74ms/step - loss: 0.8996 - accuracy: 0.6633
- val_loss: 0.9462 - val_accuracy: 0.6533
Epoch 11/100
86/86 [==============================] - 6s 74ms/step - loss: 0.8911 - accuracy: 0.6778
- val_loss: 0.8752 - val_accuracy: 0.6788
Epoch 12/100
86/86 [==============================] - 7s 76ms/step - loss: 0.7867 - accuracy: 0.7047
- val_loss: 0.8742 - val_accuracy: 0.6861
Epoch 13/100
86/86 [==============================] - 6s 75ms/step - loss: 0.7714 - accuracy: 0.7025
- val_loss: 0.8677 - val_accuracy: 0.6788
Epoch 14/100
86/86 [==============================] - 6s 75ms/step - loss: 0.6915 - accuracy: 0.7389
- val_loss: 0.6794 - val_accuracy: 0.7409
Epoch 15/100
86/86 [==============================] - 6s 73ms/step - loss: 0.7032 - accuracy: 0.7447
- val_loss: 0.7263 - val_accuracy: 0.7372
Epoch 16/100
86/86 [==============================] - 6s 73ms/step - loss: 0.6767 - accuracy: 0.7484
- val_loss: 1.1871 - val_accuracy: 0.6168
Epoch 17/100
86/86 [==============================] - 6s 73ms/step - loss: 0.6372 - accuracy: 0.7709
- val_loss: 0.6991 - val_accuracy: 0.7482
Epoch 18/100
86/86 [==============================] - 6s 75ms/step - loss: 0.6498 - accuracy: 0.7629
- val_loss: 0.6339 - val_accuracy: 0.7847
Epoch 19/100
86/86 [==============================] - 6s 73ms/step - loss: 0.5936 - accuracy: 0.7738
- val_loss: 0.7988 - val_accuracy: 0.7007
Epoch 20/100
86/86 [==============================] - 6s 75ms/step - loss: 0.5568 - accuracy: 0.7898
- val_loss: 0.5944 - val_accuracy: 0.7956
Epoch 21/100
86/86 [==============================] - 6s 73ms/step - loss: 0.5320 - accuracy: 0.8145
- val_loss: 0.6752 - val_accuracy: 0.7153
Epoch 22/100
86/86 [==============================] - 6s 73ms/step - loss: 0.4646 - accuracy: 0.8240
- val_loss: 0.7197 - val_accuracy: 0.7299
Epoch 23/100
86/86 [==============================] - 6s 73ms/step - loss: 0.4876 - accuracy: 0.8255
- val_loss: 0.7073 - val_accuracy: 0.7409
Epoch 24/100
86/86 [==============================] - 6s 75ms/step - loss: 0.4874 - accuracy: 0.8313
- val_loss: 0.5076 - val_accuracy: 0.8066
Epoch 25/100
```

```
86/86 [==============================] - 6s 74ms/step - loss: 0.4377 - accuracy: 0.8393
- val_loss: 1.1334 - val_accuracy: 0.6277
Epoch 26/100
86/86 [==============================] - 6s 74ms/step - loss: 0.4847 - accuracy: 0.8400
- val_loss: 0.8191 - val_accuracy: 0.6934
Epoch 27/100
86/86 [==============================] - 6s 74ms/step - loss: 0.4877 - accuracy: 0.8175
- val_loss: 0.7374 - val_accuracy: 0.7117
Epoch 28/100
86/86 [==============================] - 6s 75ms/step - loss: 0.3976 - accuracy: 0.8625
- val_loss: 0.3957 - val_accuracy: 0.8358
Epoch 29/100
86/86 [==============================] - 7s 78ms/step - loss: 0.3684 - accuracy: 0.8713
- val_loss: 0.4428 - val_accuracy: 0.8613
Epoch 30/100
86/86 [==============================] - 6s 74ms/step - loss: 0.3429 - accuracy: 0.8655
- val_loss: 0.5663 - val_accuracy: 0.7810
Epoch 31/100
86/86 [==============================] - 6s 74ms/step - loss: 0.3477 - accuracy: 0.8720
- val_loss: 0.9669 - val_accuracy: 0.6569
Epoch 32/100
86/86 [==============================] - 6s 74ms/step - loss: 0.4256 - accuracy: 0.8495
- val_loss: 0.5377 - val_accuracy: 0.8066
Epoch 33/100
86/86 [==============================] - 6s 75ms/step - loss: 0.3977 - accuracy: 0.8567
- val_loss: 0.4001 - val_accuracy: 0.8504
```

In [ ]:

```
score = model.evaluate(X_test,Y_test, batch_size=16)
```

```
26/26 [==============================] - 1s 20ms/step - loss: 0.3534 - accuracy: 0.8983
```

The model accuracy for test dataset is 89.83%.

ResNet50

When building the last layers of ResNet50, I first added the GlobalAveragePooling2D() to create feature map for each cagetory. I then added a dense layer. I tried different unit values and different activation functions and found that unit = 1500 and sigmoid activation improves the model performance the best. I also tried adding another dense layer but it didn't help. I tried several drop out values and found 0.4 the best. After tuning the last layers, I unfreeze the base model and retrain the whole model with a very low learning rate. I've tried some different values of learning rate and found lr = le-5 the best. When fit the model, I used EarlyStopping function in keras to find the optimal epoch value (=27) to avoid the issue of overfiffting.

In [ ]:

```
#Load the Xception pre-trained model
#include_top=False means that you’re not interested in the last layer of the model. You
base_model = keras.applications.ResNet50(
    weights='imagenet',
    input_shape=(img_height, img_width, 3),
    include_top=False)
```

In [ ]:
```
#To prevent the base model being retrained
base_model.trainable = False

inputs = keras.Input(shape=(img_height, img_width, 3))

# Preprocess inputs as expected by ResNet
x = tf.keras.applications.resnet.preprocess_input(inputs)
```

In [ ]:
```
#Build the last layers
#Use the functional API method in Keras to illustrate this approach
x = base_model(x, training=False)
x = keras.layers.GlobalAveragePooling2D()(x)
x = keras.layers.Dense(1500, activation="sigmoid")(x)
x = keras.layers.Dropout(0.4)(x)
outputs = keras.layers.Dense(10)(x)
model = keras.Model(inputs, outputs)
```

In [ ]:
```
model.summary()
```

```
Model: "model_19"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_23 (InputLayer)       [(None, 100, 100, 3)]     0

 tf.__operators__.getitem_19  (None, 100, 100, 3)      0
  (SlicingOpLambda)

 tf.nn.bias_add_19 (TFOpLamb  (None, 100, 100, 3)      0
 da)

 resnet50 (Functional)       (None, 4, 4, 2048)        23587712

 global_average_pooling2d_19  (None, 2048)             0
  (GlobalAveragePooling2D)

 dense_37 (Dense)            (None, 1500)              3073500

 dropout_19 (Dropout)        (None, 1500)              0

 dense_38 (Dense)            (None, 10)                15010

=================================================================
Total params: 26,676,222
Trainable params: 3,088,510
Non-trainable params: 23,587,712
_________________________________________________________________
```

In [ ]:
```
model.compile(optimizer='adam',
              loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
model.fit(X, Y, epochs=3, validation_data=(X_test,Y_test))
```

```
Epoch 1/3
52/52 [==============================] - 6s 58ms/step - loss: 2.6627 - accuracy: 0.1019
- val_loss: 2.3031 - val_accuracy: 0.1332
Epoch 2/3
52/52 [==============================] - 2s 40ms/step - loss: 2.4126 - accuracy: 0.1358
- val_loss: 2.2197 - val_accuracy: 0.1985
Epoch 3/3
52/52 [==============================] - 2s 44ms/step - loss: 2.2966 - accuracy: 0.1625
- val_loss: 2.1997 - val_accuracy: 0.1743
```

Out[ ]: <keras.callbacks.History at 0x7fc5fdb0a400>

In [ ]:
```
# fine-tuning
base_model.trainable = True
model.summary()

model.compile(
    optimizer=keras.optimizers.Adam(1e-5),  # Low learning rate
    loss=keras.losses.CategoricalCrossentropy(from_logits=True),
    metrics=['accuracy']
)
```

```
Model: "model_19"

_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_23 (InputLayer)       [(None, 100, 100, 3)]     0         
                                                                 
 tf.__operators__.getitem_19  (None, 100, 100, 3)      0         
  (SlicingOpLambda)                                              
                                                                 
 tf.nn.bias_add_19 (TFOpLamb  (None, 100, 100, 3)      0         
 da)                                                             
                                                                 
 resnet50 (Functional)       (None, 4, 4, 2048)        23587712  
                                                                 
 global_average_pooling2d_19  (None, 2048)             0         
  (GlobalAveragePooling2D)                                       
                                                                 
 dense_37 (Dense)            (None, 1500)              3073500   
                                                                 
 dropout_19 (Dropout)        (None, 1500)              0         
                                                                 
 dense_38 (Dense)            (None, 10)                15010     
                                                                 
=================================================================
Total params: 26,676,222
Trainable params: 26,623,102
Non-trainable params: 53,120
_________________________________________________________________
```

In [ ]:
```
from keras import callbacks
earlystopping = callbacks.EarlyStopping(monitor ="val_loss",
                                        mode ="min", patience = 5,
                                        restore_best_weights = True)

history = model.fit(partial, partial_labels, batch_size = 16,
                    epochs = 100, validation_data =(val, val_labels),
                    callbacks =[earlystopping])
```

```
Epoch 1/100
86/86 [==============================] - 13s 89ms/step - loss: 2.2641 - accuracy: 0.1702
- val_loss: 1.9824 - val_accuracy: 0.3869
Epoch 2/100
86/86 [==============================] - 6s 73ms/step - loss: 1.8381 - accuracy: 0.3185
- val_loss: 1.5089 - val_accuracy: 0.4672
Epoch 3/100
86/86 [==============================] - 6s 72ms/step - loss: 1.3676 - accuracy: 0.4815
- val_loss: 1.4244 - val_accuracy: 0.4307
Epoch 4/100
86/86 [==============================] - 7s 76ms/step - loss: 1.1295 - accuracy: 0.5615
- val_loss: 1.1015 - val_accuracy: 0.5839
Epoch 5/100
```

```
86/86 [==============================] - 6s 73ms/step - loss: 0.8810 - accuracy: 0.6618
- val_loss: 0.7937 - val_accuracy: 0.7263
Epoch 6/100
86/86 [==============================] - 6s 72ms/step - loss: 0.8137 - accuracy: 0.6851
- val_loss: 1.0521 - val_accuracy: 0.6314
Epoch 7/100
86/86 [==============================] - 6s 74ms/step - loss: 0.7253 - accuracy: 0.7265
- val_loss: 0.6256 - val_accuracy: 0.8066
Epoch 8/100
86/86 [==============================] - 6s 75ms/step - loss: 0.5901 - accuracy: 0.7927
- val_loss: 0.7293 - val_accuracy: 0.7628
Epoch 9/100
86/86 [==============================] - 6s 72ms/step - loss: 0.5708 - accuracy: 0.8015
- val_loss: 0.6339 - val_accuracy: 0.7664
Epoch 10/100
86/86 [==============================] - 6s 71ms/step - loss: 0.4804 - accuracy: 0.8167
- val_loss: 0.6785 - val_accuracy: 0.7737
Epoch 11/100
86/86 [==============================] - 6s 71ms/step - loss: 0.4195 - accuracy: 0.8480
- val_loss: 0.6480 - val_accuracy: 0.7774
Epoch 12/100
86/86 [==============================] - 6s 73ms/step - loss: 0.3681 - accuracy: 0.8655
- val_loss: 0.4904 - val_accuracy: 0.8467
Epoch 13/100
86/86 [==============================] - 6s 72ms/step - loss: 0.4440 - accuracy: 0.8349
- val_loss: 0.4252 - val_accuracy: 0.8467
Epoch 14/100
86/86 [==============================] - 6s 72ms/step - loss: 0.3258 - accuracy: 0.8887
- val_loss: 0.3766 - val_accuracy: 0.8796
Epoch 15/100
86/86 [==============================] - 6s 71ms/step - loss: 0.2876 - accuracy: 0.8938
- val_loss: 0.5750 - val_accuracy: 0.8066
Epoch 16/100
86/86 [==============================] - 6s 71ms/step - loss: 0.2602 - accuracy: 0.9047
- val_loss: 0.6998 - val_accuracy: 0.7664
Epoch 17/100
86/86 [==============================] - 6s 72ms/step - loss: 0.3394 - accuracy: 0.8807
- val_loss: 0.3426 - val_accuracy: 0.8686
Epoch 18/100
86/86 [==============================] - 6s 73ms/step - loss: 0.2800 - accuracy: 0.8967
- val_loss: 0.3222 - val_accuracy: 0.8978
Epoch 19/100
86/86 [==============================] - 6s 71ms/step - loss: 0.2023 - accuracy: 0.9295
- val_loss: 0.3296 - val_accuracy: 0.8686
Epoch 20/100
86/86 [==============================] - 6s 75ms/step - loss: 0.2439 - accuracy: 0.9193
- val_loss: 0.3507 - val_accuracy: 0.8540
Epoch 21/100
86/86 [==============================] - 6s 71ms/step - loss: 0.2259 - accuracy: 0.9244
- val_loss: 0.4198 - val_accuracy: 0.8504
Epoch 22/100
86/86 [==============================] - 6s 72ms/step - loss: 0.1532 - accuracy: 0.9455
- val_loss: 0.2568 - val_accuracy: 0.9015
Epoch 23/100
86/86 [==============================] - 6s 71ms/step - loss: 0.2824 - accuracy: 0.8975
- val_loss: 0.4117 - val_accuracy: 0.8431
Epoch 24/100
86/86 [==============================] - 6s 73ms/step - loss: 0.1467 - accuracy: 0.9469
- val_loss: 0.3129 - val_accuracy: 0.8796
Epoch 25/100
86/86 [==============================] - 6s 71ms/step - loss: 0.1547 - accuracy: 0.9462
- val_loss: 0.9101 - val_accuracy: 0.7080
Epoch 26/100
86/86 [==============================] - 6s 72ms/step - loss: 0.1933 - accuracy: 0.9302
- val_loss: 0.3889 - val_accuracy: 0.8796
```

```
Epoch 27/100
86/86 [==============================] - 6s 74ms/step - loss: 0.2016 - accuracy: 0.9375
- val_loss: 0.3725 - val_accuracy: 0.8577
```

In [ ]:
```
score = model.evaluate(X_test,Y_test, batch_size=16)
```

```
26/26 [==============================] - 1s 23ms/step - loss: 0.2531 - accuracy: 0.9225
```

The model accuracy for test dataset is 92.25%.

Inceptionv3

When building the last layers of Inceptionv3, I first added the GlobalAveragePooling2D() to create feature map for each cagetory. I then added a dense layer of 1200 units with relu activation and found model performance improved. I tried other activation methods and model didn't improve. I also tried adding another dense layer and performance decreased. I tried several drop out values and found 0.3 the best. After tuning the last layers, I unfreeze the base model and retrain the whole model with a very low learning rate. I've tried some different values of learning rate and found lr = le-6 the best. When fit the model, I used EarlyStopping function in keras to find the optimal epoch value to avoid the issue of overfiffting.

In [ ]:
```
seed(123)
tf.random.set_seed(123)
```

In [ ]:
```
#Load the Xception pre-trained model
#include_top=False means that you’re not interested in the last layer of the model. You
base_model = keras.applications.InceptionV3(
    weights='imagenet',
    input_shape=(img_height, img_width, 3),
    include_top=False)
```

In [ ]:
```
#To prevent the base model being retrained
base_model.trainable = False

inputs = keras.Input(shape=(img_height, img_width, 3))

# Preprocess inputs as expected by ResNet
x = tf.keras.applications.inception_v3.preprocess_input(inputs)
```

In [ ]:
```
#Build the last layers
#Use the functional API method in Keras to illustrate this approach
x = base_model(x, training=False)
x = keras.layers.GlobalAveragePooling2D()(x)
x = keras.layers.Dense(1200, activation="relu")(x)
x = keras.layers.Dropout(0.3)(x)
outputs = keras.layers.Dense(10)(x)
model = keras.Model(inputs, outputs)
```

In [ ]:
```
model.compile(optimizer='adam',
              loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
model.fit(X, Y, epochs=3, validation_data=(X_test,Y_test))
score = model.evaluate(X_test,Y_test, batch_size=16)
```

```
Epoch 1/3
52/52 [==============================] - 7s 51ms/step - loss: 1.8960 - accuracy: 0.3505
```

```
- val_loss: 1.3733 - val_accuracy: 0.4843
Epoch 2/3
52/52 [==============================] - 1s 26ms/step - loss: 1.2632 - accuracy: 0.5561
- val_loss: 1.0339 - val_accuracy: 0.6852
Epoch 3/3
52/52 [==============================] - 1s 26ms/step - loss: 1.0522 - accuracy: 0.6295
- val_loss: 0.9495 - val_accuracy: 0.6925
26/26 [==============================] - 0s 17ms/step - loss: 0.9495 - accuracy: 0.6925
```

In [ ]:

```
base_model.trainable = True
model.summary()

model.compile(
    optimizer=keras.optimizers.Adam(1e-5),  # Low learning rate
    loss=keras.losses.CategoricalCrossentropy(from_logits=True),
    metrics=['accuracy']
)
```

```
Model: "model_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_26 (InputLayer)       [(None, 100, 100, 3)]     0         
                                                                 
 tf.math.truediv_12 (TFOpLam  (None, 100, 100, 3)      0         
 bda)                                                            
                                                                 
 tf.math.subtract_12 (TFOpLa  (None, 100, 100, 3)      0         
 mbda)                                                           
                                                                 
 inception_v3 (Functional)   (None, 1, 1, 2048)        21802784  
                                                                 
 global_average_pooling2d_12  (None, 2048)             0         
  (GlobalAveragePooling2D)                                       
                                                                 
 dense_25 (Dense)            (None, 1200)              2458800   
                                                                 
 dropout_12 (Dropout)        (None, 1200)              0         
                                                                 
 dense_26 (Dense)            (None, 10)                12010     
                                                                 
=================================================================
Total params: 24,273,594
Trainable params: 24,239,162
Non-trainable params: 34,432
_________________________________________________________________
```

In [ ]:

```
from keras import callbacks
earlystopping = callbacks.EarlyStopping(monitor ="val_loss",
                                        mode ="min", patience = 5,
                                        restore_best_weights = True)

history = model.fit(partial, partial_labels, batch_size = 16,
                    epochs = 100, validation_data =(val, val_labels),
                    callbacks =[earlystopping])
```

```
Epoch 1/100
52/52 [==============================] - 11s 101ms/step - loss: 2.0680 - accuracy: 0.297
0 - val_loss: 1.4040 - val_accuracy: 0.5267
Epoch 2/100
52/52 [==============================] - 3s 61ms/step - loss: 1.4204 - accuracy: 0.4691
- val_loss: 1.5199 - val_accuracy: 0.4041
Epoch 3/100
52/52 [==============================] - 3s 64ms/step - loss: 1.2795 - accuracy: 0.4824
```

```
- val_loss: 1.1156 - val_accuracy: 0.5692
Epoch 4/100
52/52 [==============================] - 3s 64ms/step - loss: 1.1120 - accuracy: 0.5673
- val_loss: 1.0592 - val_accuracy: 0.6129
Epoch 5/100
52/52 [==============================] - 3s 64ms/step - loss: 0.8865 - accuracy: 0.6618
- val_loss: 0.9533 - val_accuracy: 0.5934
Epoch 6/100
52/52 [==============================] - 3s 64ms/step - loss: 1.1001 - accuracy: 0.5976
- val_loss: 0.7834 - val_accuracy: 0.7403
Epoch 7/100
52/52 [==============================] - 3s 64ms/step - loss: 0.7892 - accuracy: 0.7139
- val_loss: 0.6755 - val_accuracy: 0.7367
Epoch 8/100
52/52 [==============================] - 4s 72ms/step - loss: 0.6774 - accuracy: 0.7394
- val_loss: 0.8975 - val_accuracy: 0.6917
Epoch 9/100
52/52 [==============================] - 4s 71ms/step - loss: 0.6128 - accuracy: 0.7782
- val_loss: 0.6495 - val_accuracy: 0.7536
Epoch 10/100
52/52 [==============================] - 4s 78ms/step - loss: 0.6842 - accuracy: 0.7515
- val_loss: 0.5993 - val_accuracy: 0.7803
Epoch 11/100
52/52 [==============================] - 4s 75ms/step - loss: 0.5318 - accuracy: 0.8194
- val_loss: 0.5760 - val_accuracy: 0.7864
Epoch 12/100
52/52 [==============================] - 3s 65ms/step - loss: 0.3998 - accuracy: 0.8739
- val_loss: 0.4600 - val_accuracy: 0.8374
Epoch 13/100
52/52 [==============================] - 4s 72ms/step - loss: 0.4576 - accuracy: 0.8509
- val_loss: 0.7318 - val_accuracy: 0.7342
Epoch 14/100
52/52 [==============================] - 3s 65ms/step - loss: 0.7270 - accuracy: 0.7406
- val_loss: 0.4524 - val_accuracy: 0.8556
Epoch 15/100
52/52 [==============================] - 4s 72ms/step - loss: 0.4532 - accuracy: 0.8424
- val_loss: 0.6309 - val_accuracy: 0.7791
Epoch 16/100
52/52 [==============================] - 4s 72ms/step - loss: 0.4121 - accuracy: 0.8473
- val_loss: 0.5096 - val_accuracy: 0.8228
Epoch 17/100
52/52 [==============================] - 3s 64ms/step - loss: 0.4315 - accuracy: 0.8364
- val_loss: 0.3523 - val_accuracy: 0.8774
Epoch 18/100
52/52 [==============================] - 3s 64ms/step - loss: 0.2602 - accuracy: 0.9079
- val_loss: 0.3145 - val_accuracy: 0.8883
Epoch 19/100
52/52 [==============================] - 3s 62ms/step - loss: 0.2696 - accuracy: 0.8958
- val_loss: 0.3883 - val_accuracy: 0.8726
Epoch 20/100
52/52 [==============================] - 3s 61ms/step - loss: 0.2833 - accuracy: 0.9030
- val_loss: 0.3617 - val_accuracy: 0.8750
Epoch 21/100
52/52 [==============================] - 3s 61ms/step - loss: 0.3654 - accuracy: 0.8715
- val_loss: 0.3713 - val_accuracy: 0.8629
Epoch 22/100
52/52 [==============================] - 3s 62ms/step - loss: 0.3447 - accuracy: 0.8836
- val_loss: 0.3816 - val_accuracy: 0.8617
Epoch 23/100
52/52 [==============================] - 3s 64ms/step - loss: 0.3965 - accuracy: 0.8545
- val_loss: 0.3652 - val_accuracy: 0.8726
```

In [ ]:

```
score = model.evaluate(X_test,Y_test, batch_size=16)
```

```
26/26 [==============================] - 1s 21ms/step - loss: 0.3001 - accuracy: 0.9007
```

The Inceptionv3 model has a accuracy of 90.07%.

1. Model Comparison

Model	**Accuracy**
VGG-16 - University of Oxford	98.73%
ResNet -50 - Microsoft	92.25%
InceptionV3 - Google	90.07%
X-ception - Google	89.83%
EfficientNetB0 - Google	83.99%
CNN	80.34%
Fully Connected Structure	71.91%